Quarkus Grundlagen

Java Dev-Ops Cloud Training

11 - 2021

Inhalt

- Quarkus – Voraussetzungen - Prerequsites
- Maven Quickstart: Create by hand
- Quarkus Quickstart: create via IntelliJ
- Quickstart : launch new app
- Quarkus – Generated Output
- Quarkus Maven plugin
- Start Quarkus App
- Quarkus – Rest API
- Add to git
- Import into intelliJ
- IntellJ Project
- Maven Settings in IntelliJ
- Intellij: Java Settings for Java Project
- Run Quarkus from IntellJ maven
- Quarkus Development Mode
- IntelliJ - Quarkus – Attach Debugger
- Quarkus - Testing

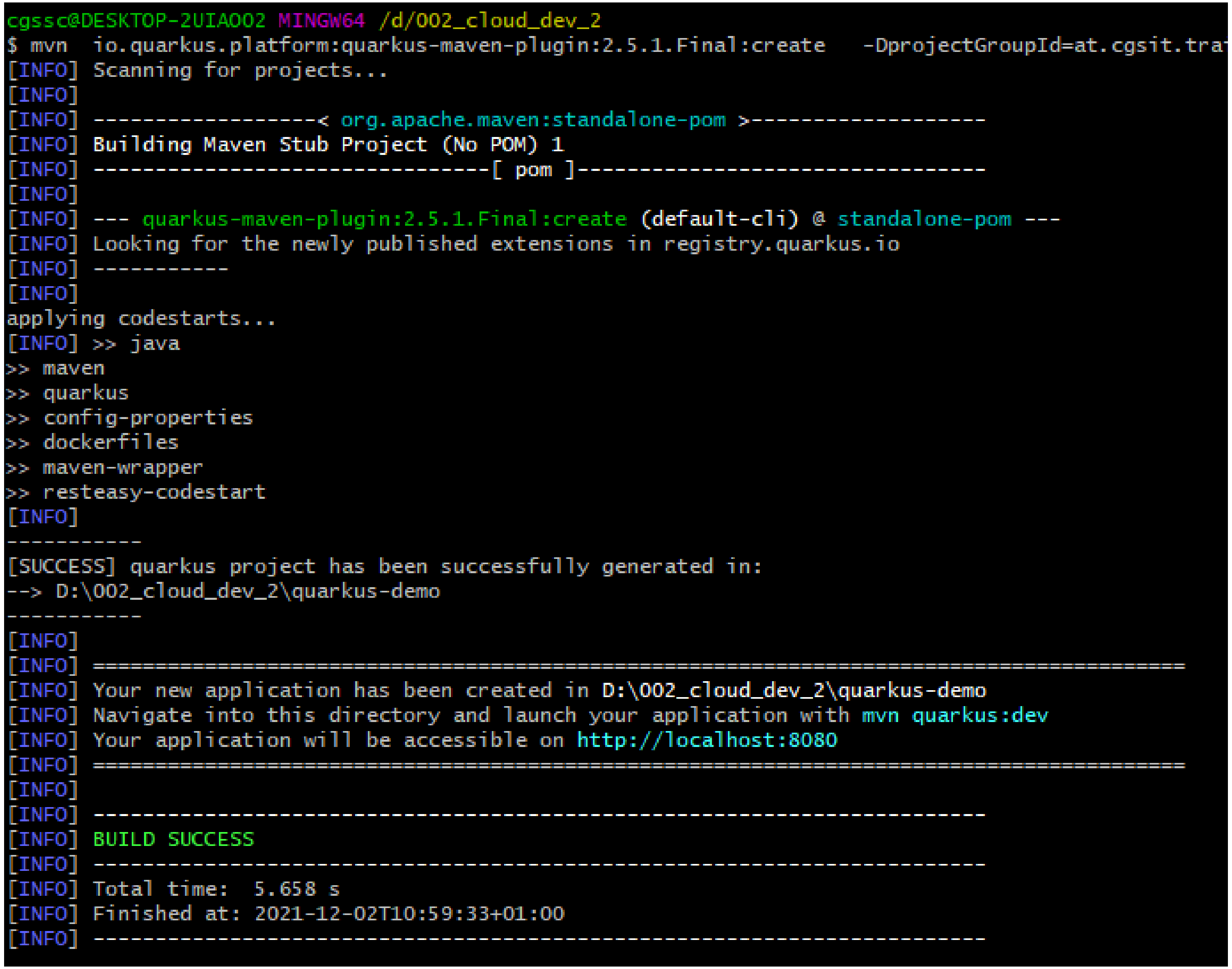
Quarkus – Voraussetzungen - Prerequsites

- IDE: IntelliJ or Eclipse
- JDK 11+ installed with JAVA_HOME configured appropriately
- Apache Maven 3.8.1+

Maven Quickstart: Create by hand

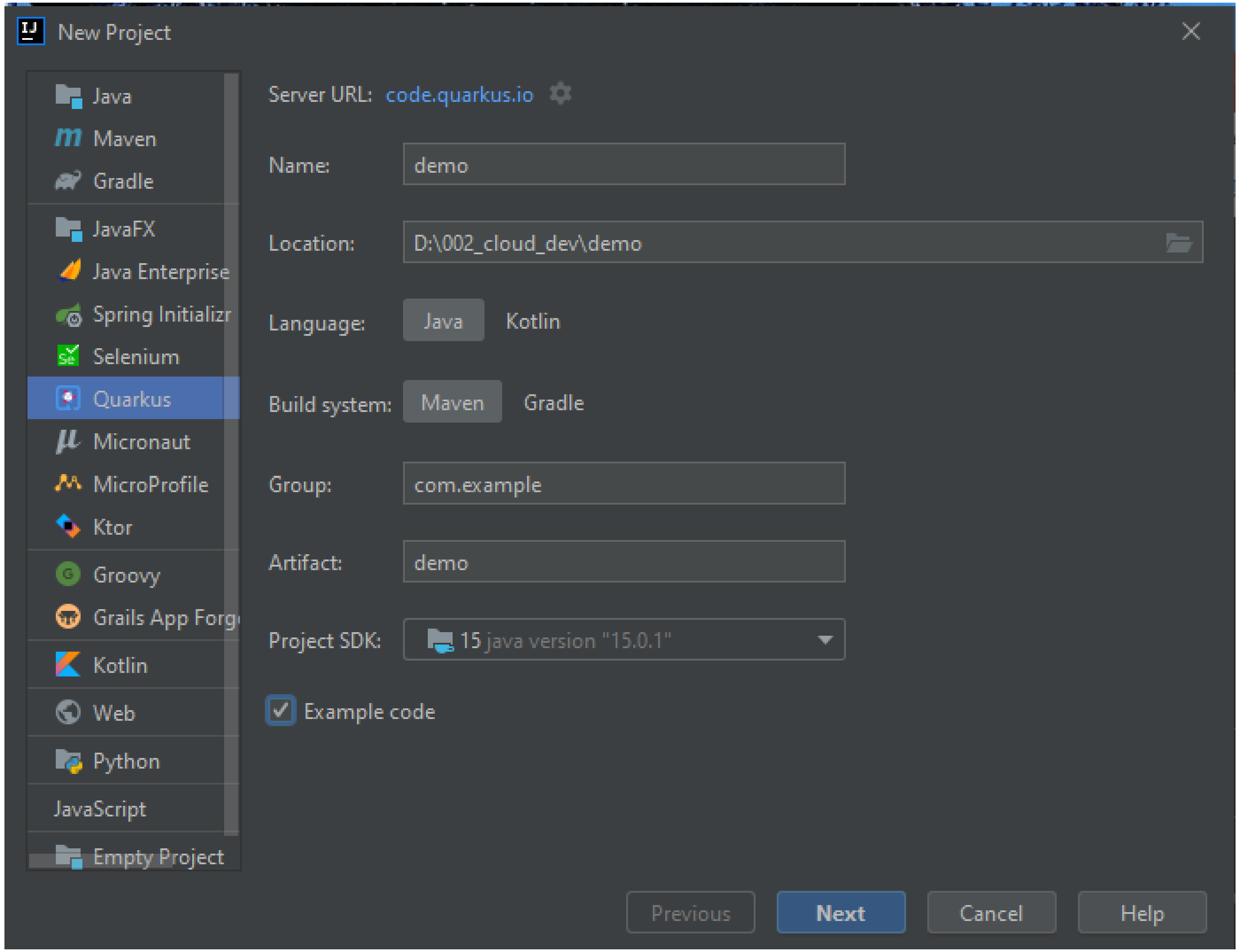
```
mvn io.quarkus.platform:quarkus-maven-plugin:2.5.1.Final:create \
    -DprojectGroupId=at.cgsit.train \
    -DprojectArtifactId=quarkus-demo \
    -DclassName="at.cgsit.train.quarkus.DemoResource" \
    -Dpath="/helloDemo„

cd quarkus-demo
```

Build success



```
cgssc@DESKTOP-2UIA002 MINGW64 /d/002_cloud_dev_2
$ mvn  io.quarkus.platform:quarkus-maven-plugin:2.5.1.Final:create   -DprojectGroupId=at.cgsit.tra
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] --- quarkus-maven-plugin:2.5.1.Final:create (default-cli) @ standalone-pom ---
[INFO] Looking for the newly published extensions in registry.quarkus.io
[INFO] -----------
[INFO]
applying codestarts...
[INFO] >> java
>> maven
>> quarkus
>> config-properties
>> dockerfiles
>> maven-wrapper
>> resteasy-codestart
[INFO]
-----------
[SUCCESS] quarkus project has been successfully generated in:
--> D:\002_cloud_dev_2\quarkus-demo
-----------
[INFO]
[INFO] ========================================================================================
[INFO] Your new application has been created in D:\002_cloud_dev_2\quarkus-demo
[INFO] Navigate into this directory and launch your application with mvn quarkus:dev
[INFO] Your application will be accessible on http://localhost:8080
[INFO] ========================================================================================
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.658 s
[INFO] Finished at: 2021-12-02T10:59:33+01:00
[INFO] ------------------------------------------------------------------------
```

Quarkus Quickstart: create via IntelliJ



Server URL: code.quarkus.io

Name: demo

Location: D:\002_cloud_dev\demo

Language: Java Kotlin

Build system: Maven Gradle

Group: com.example

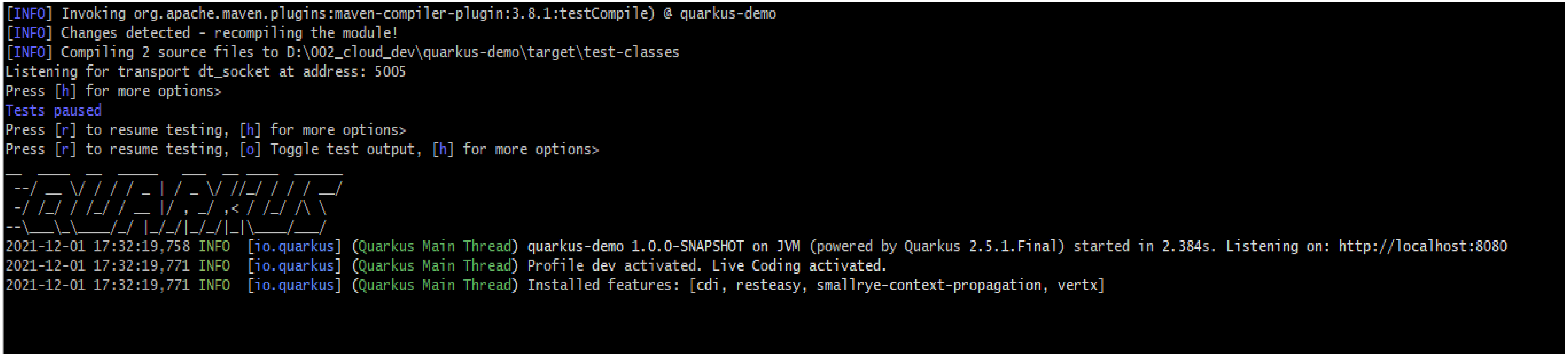
Artifact: demo

Project SDK: 15 java version "15.0.1"

☑ Example code

Quickstart : launch new app

- cd quarkus-demo
- mvn quarkus:dev
- Browser: http://localhost:5005/

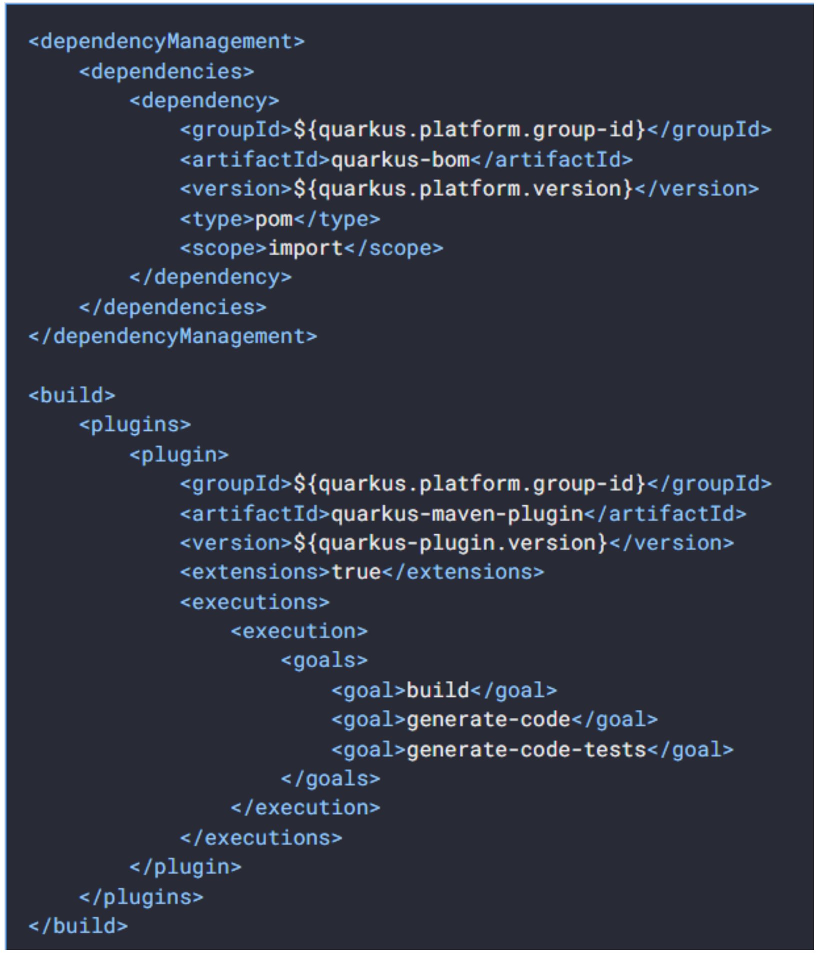


```
[INFO] Invoking org.apache.maven.plugins:maven-compiler-plugin:3.8.1:testCompile) @ quarkus-demo
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to D:\002_cloud_dev\quarkus-demo\target\test-classes
Listening for transport dt_socket at address: 5005
Press [h] for more options>
Tests paused
Press [r] to resume testing, [h] for more options>
Press [r] to resume testing, [o] Toggle test output, [h] for more options>
__  ____  __  _____   ___  __ ____  ______ 
 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/ 
 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \   
--\___\_\____/_/ |_/_/|_/_/|_|\____/___/   
2021-12-01 17:32:19,758 INFO  [io.quarkus] (Quarkus Main Thread) quarkus-demo 1.0.0-SNAPSHOT on JVM (powered by Quarkus 2.5.1.Final) started in 2.384s. Listening on: http://localhost:8080
2021-12-01 17:32:19,771 INFO  [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2021-12-01 17:32:19,771 INFO  [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, resteasy, smallrye-context-propagation, vertx]
```

Quarkus – Generated Output

- the Maven structure
- an org.acme.getting.started.GreetingResource resource exposed on /hello
- an associated unit test
- a landing page that is accessible on http://localhost:8080 after starting the application
- example Dockerfile files for both native and jvm modes in src/main/docker
- the application configuration file

Quarkus Maven plugin



```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>${quarkus.platform.group-id}</groupId>
            <artifactId>quarkus-bom</artifactId>
            <version>${quarkus.platform.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<build>
    <plugins>
        <plugin>
            <groupId>${quarkus.platform.group-id}</groupId>
            <artifactId>quarkus-maven-plugin</artifactId>
            <version>${quarkus-plugin.version}</version>
            <extensions>true</extensions>
            <executions>
                <execution>
                    <goals>
                        <goal>build</goal>
                        <goal>generate-code</goal>
                        <goal>generate-code-tests</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

- Quarkus BOM, allowing you to omit the version on the different Quarkus dependencies

- quarkus-maven-plugin responsible of the packaging of the application and also providing the development mode

Start Quarkus App

- Edit Rest Service
- Auto-Deployment is done automatically
- Open w and d in quarkus command line console

Or use

curl -w "\n" http://localhost:8080/helloDemo

Quarkus – Rest API

Differences with vanilla JAX-RS

1. With Quarkus, there is no need to create an Application class.
   It’s supported, but not required.
2. In addition, only one instance of the resource is created and not one per request.
3. You can configure this using the different
   *Scoped annotations (ApplicationScoped, RequestScoped, etc).

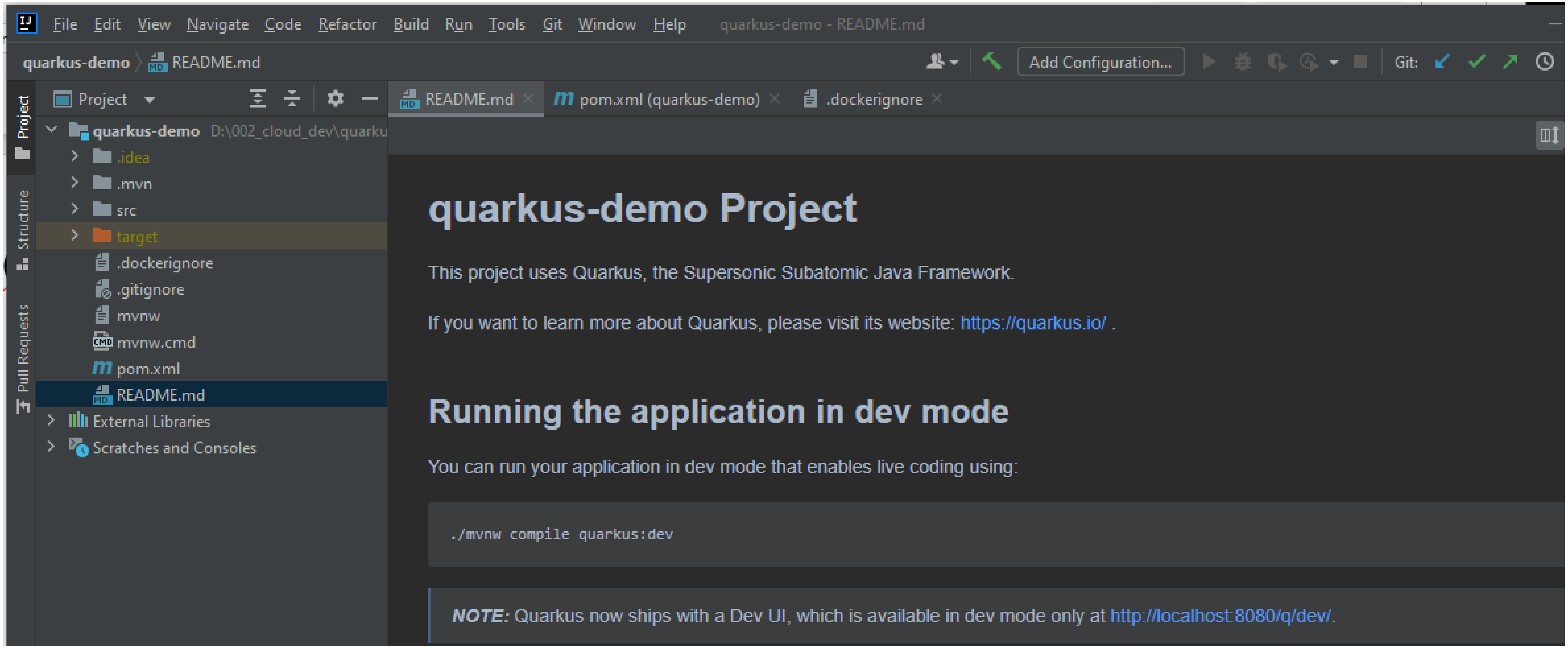
Add to git

```
git init
git add –A
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/chrascher/quarkus_demo.git

git remote add origin https://github.com/<user>/quarkus_demo.git

git push -u origin main

git pull
```

Import into intelliJ

- Open IntelliJ → Open Project
- Choose directory: z.b. D:\002_cloud_dev\quarkus-demo
- Open project
- Conntect to github and provide password

IntellJ Project



quarkus-demo Project

This project uses Quarkus, the Supersonic Subatomic Java Framework.

If you want to learn more about Quarkus, please visit its website: https://quarkus.io/ .

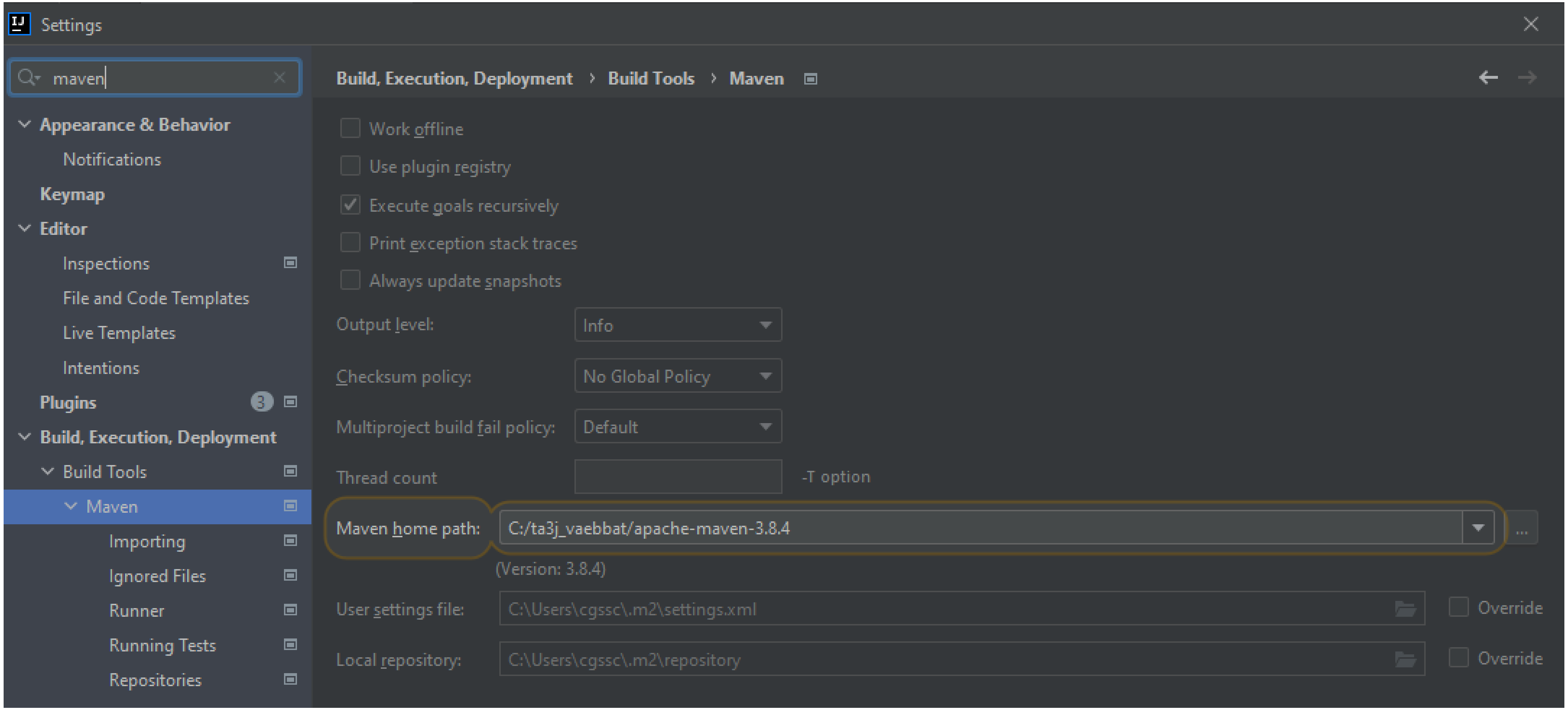
Running the application in dev mode

You can run your application in dev mode that enables live coding using:

```
./mvnw compile quarkus:dev
```

> ***NOTE:*** Quarkus now ships with a Dev UI, which is available in dev mode only at http://localhost:8080/q/dev/.

Maven Settings in IntelliJ



Build, Execution, Deployment › Build Tools › Maven

□ Work offline

□ Use plugin registry

☑ Execute goals recursively

□ Print exception stack traces

□ Always update snapshots

Output level: Info

Checksum policy: No Global Policy

Multiproject build fail policy: Default

Thread count -T option

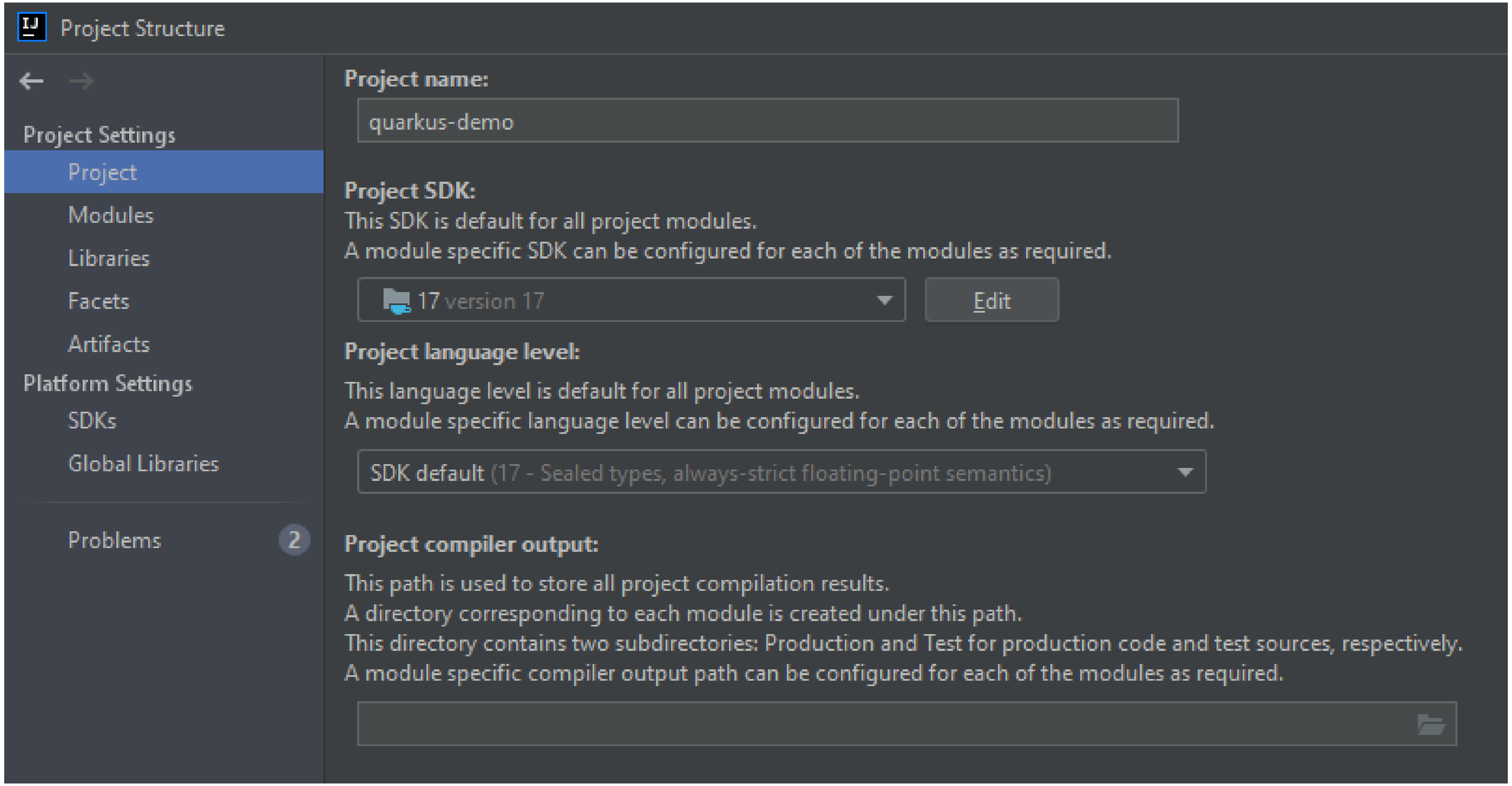
Maven home path: C:/ta3j_vaebbat/apache-maven-3.8.4

(Version: 3.8.4)

User settings file: C:\Users\cgssc\.m2\settings.xml □ Override

Local repository: C:\Users\cgssc\.m2\repository □ Override

Intellij: Java Settings for Java Project



Project name:

quarkus-demo

Project SDK:

This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.

17 version 17

Project language level:

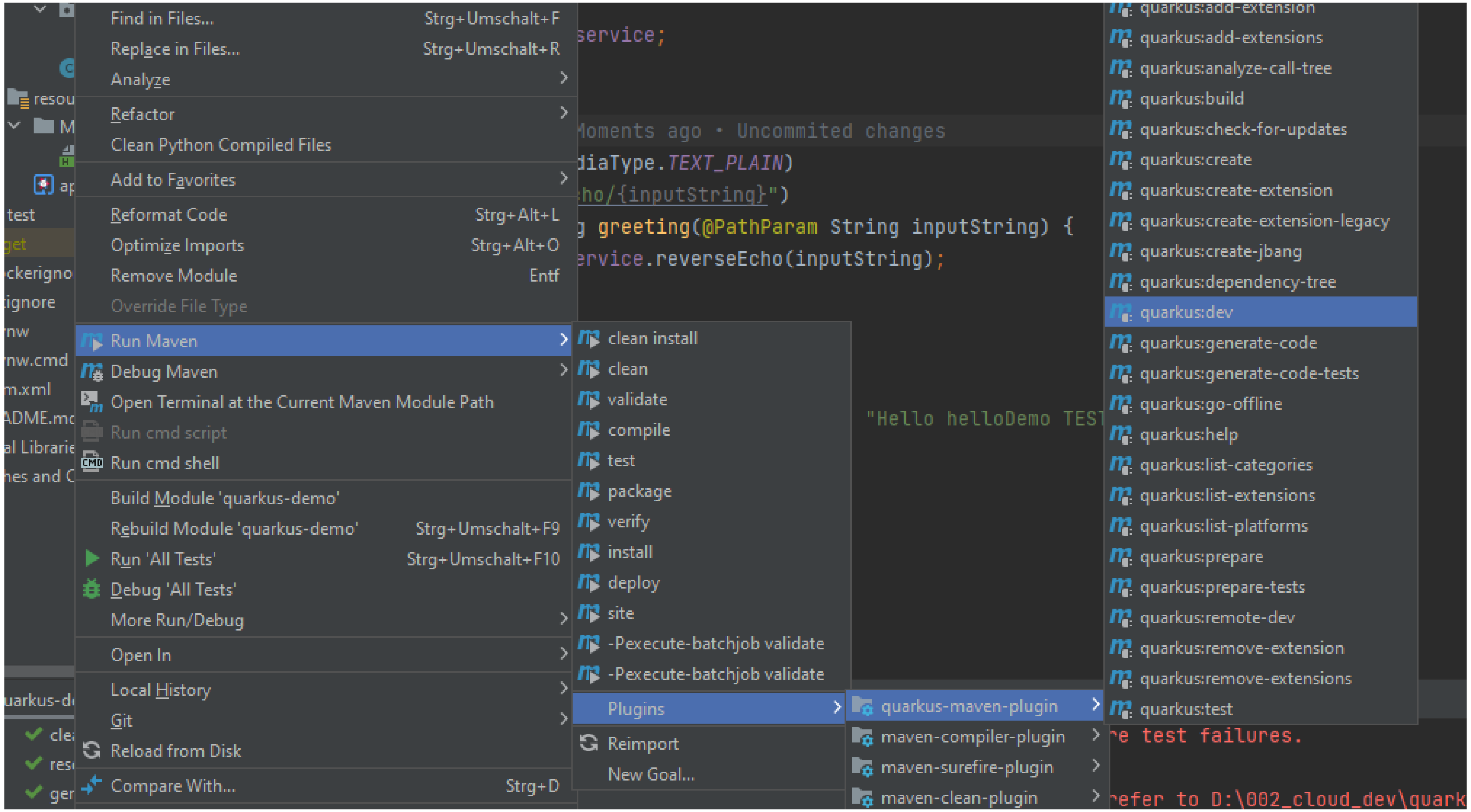
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.

SDK default (17 - Sealed types, always-strict floating-point semantics)

Project compiler output:

This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.

Run Quarkus from IntellJ maven



```
service;
diaType.TEXT_PLAIN)
ho/{inputString}")
g greeting(@PathParam String inputString) {
ervice.reverseEcho(inputString);
"Hello helloDemo TEST
e test failures.
efer to D:\002_cloud_dev\quark
```

Mvn `compile quarkus:dev`

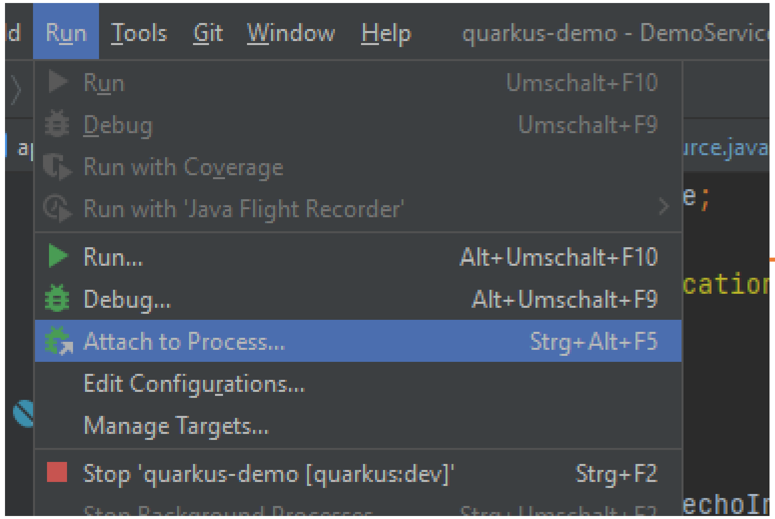
CGS ITSolutions

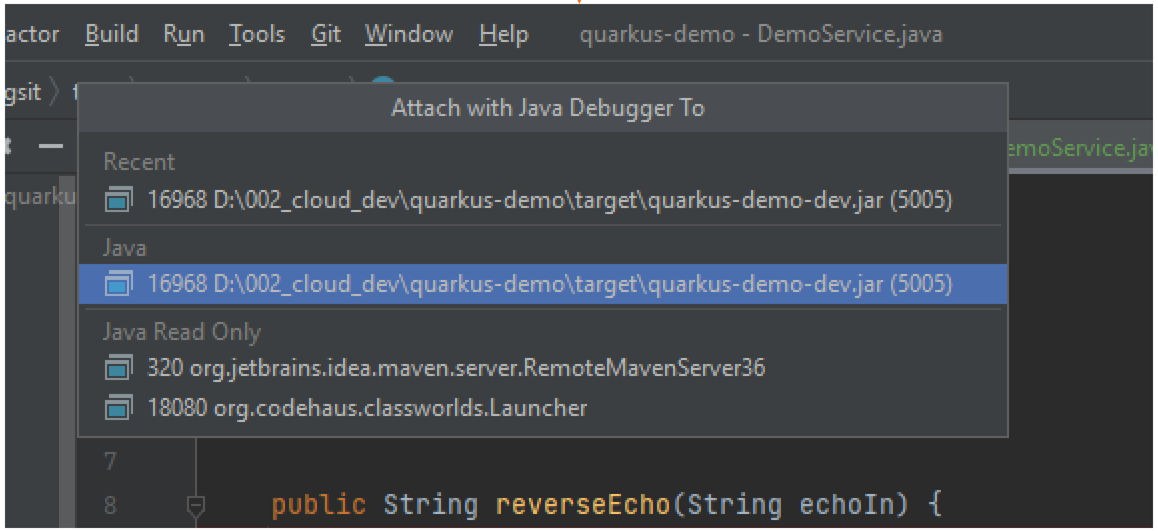
Quarkus Development Mode

quarkus:dev runs Quarkus in development mode

- This enables hot deployment with background compilation
- This works too for resource files like the configuration property file
- Refreshing the browser triggers a scan of the workspace
- This will also listen for a debugger on port 5005

IntelliJ - Quarkus – Attach Debugger



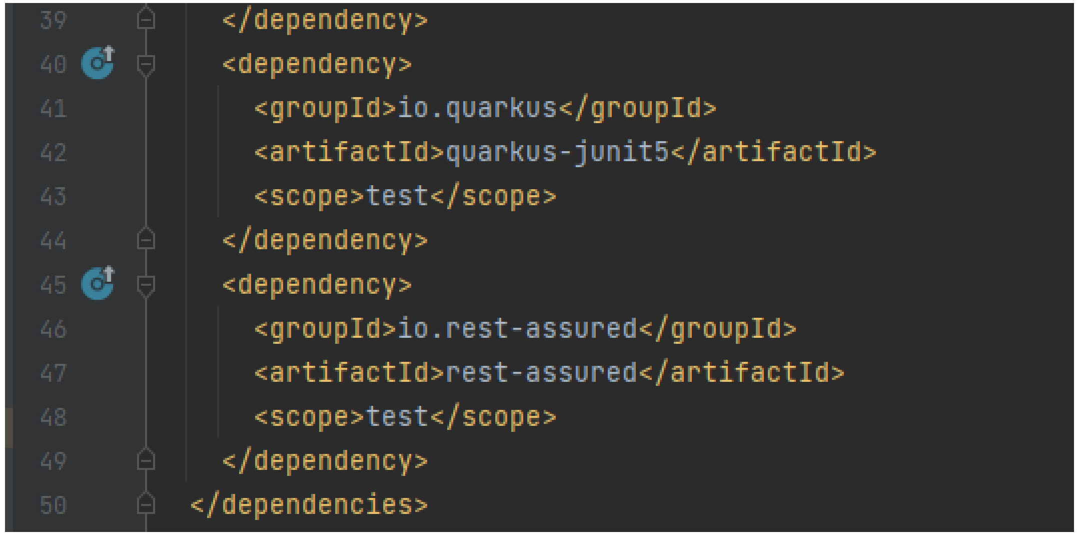


```
16968 D:\002_cloud_dev\quarkus-demo\target\quarkus-demo-dev.jar (5005)
16968 D:\002_cloud_dev\quarkus-demo\target\quarkus-demo-dev.jar (5005)
320 org.jetbrains.idea.maven.server.RemoteMavenServer36
18080 org.codehaus.classworlds.Launcher
```

```java
public String reverseEcho(String echoIn) {
```

Quarkus - Testing

In the generated pom.xml file, you can see 2 test dependencies:



```
    </dependency>
    <dependency>
      <groupId>io.quarkus</groupId>
      <artifactId>quarkus-junit5</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>io.rest-assured</groupId>
      <artifactId>rest-assured</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

Danke für Ihre Aufmerksamkeit